The Report Committee for Cyril Jose
Certifies that this is the approved version of the following report:

Inquest: System to Utilize Perceptible Information to Affirm User Identity on Personal Mobile Devices

APPROVED BY
SUPERVISING COMMITTEE:

Supervisor:

Suzanne Barber

Alaric Silveira

Inquest: System to Utilize Perceptible Information to Affirm User Identity on Personal Mobile Devices

by

Cyril Jose, B.Tech

Report

Presented to the Faculty of the Graduate School of

The University of Texas at Austin

in Partial Fulfillment

of the Requirements

for the Degree of

Master of Science in Engineering

The University of Texas at Austin

December 2015

Dedication

For

Marina, Joseph and Thomas

Acknowledgements

I would like to remember and acknowledge the support from my professors and staff at the University of Texas at Austin who have gracefully provided their time and guidance over the past few years. Professor Barber has been particularly consequential and helpful through her guidance during the preparation of this report.

I also would like to express my gratitude towards Alaric Silveira for agreeing to be the reader for this report.

Abstract

Inquest: System to Utilize Perceptible Information to Affirm User Identity on Personal Mobile Devices

Cyril Jose, MSE

The University of Texas at Austin, 2015

Supervisor: Suzanne Barber

Inquest is a secondary factor method for affirming user identity on a personal handheld computing device. The traditional personal computer market of notebooks and desk-based computers are on a steady decline and it is expected that tablets and smartphones shall complement and gradually replace desk based computers as the preferred medium for critical and sensitive personal and professional data access and computing needs. This introduces a new set of challenges quite different from traditionally ones to secure access to these personal computing devices.

This work and the report looks into aspects of affirming user identity on mobile devices making use of perceptible information readily available on that device. Inquest, the instantiation of the envisioned system and method uses a client-server model where the clients are these mobile devices listening and reporting perceptible information data crumbs to the server making use of a RESTful interface model. Predictive analytical methods are employed and any association or relationships relevant for available information is computed and stored at server to be retrieved for notifying identity

affirmation status. Data model used for storing preprocessed information is defined and effectiveness of selected predictive analysis methods against the data model is documented. The approach leveraged for generating training data is documented.

Table of Contents

List of Tables

List of Figures

Chapter 1: Introduction

1.1 VISION

Handheld devices like tablets and smartphones are increasingly becoming ubiquitous in personal computing needs. Worldwide devices (the combined shipment of PCs, tablets and mobile phones) shipment are on a pace to total at least 3 billion units a year by 2017. While there will be some individuals who retain both a personal PC and a tablet, especially those who use either or both for work and play, most will be satisfied with the experience they get from a tablet or smartphone as their main computing device [1].

The traditional personal computer market of notebooks and desk-based computers is on a steady decline and it is expected that tablets and smartphones shall complement and gradually replace desk based computers as the preferred medium for critical and sensitive personal and professional data access and computing needs. This introduces a new set of challenges quite different from traditionally ones to secure access to these personal computing devices. Desk based systems are tied to some physical location and often physical access to that device itself was a reasonable form of primary authentication. Along with user password protection and various surveillance mechanisms (e.g. video surveillance) it provided a high standard of reliable user identity enforcements in these traditional environments. Personal devices like smartphones or tablets are expected to move around along with its user most often where ever he or she goes.

This report looks into aspects of securing user identity on mobile devices making use of perceptible information readily available for that device. The system or application

software running on that device shall listen for and collect that information to be pre-processed and utilized for ensuring and validating identity claimed by that user as a method for continuous user identity affirmation.

1.2 INQUEST

Inquest is an instantiation of the above envisioned system and method. It looks into answering how such a system shall be designed and built and also explores how the system shall be used.

Inquest is modeled as a service available in the system to be enabled by the respective user for the benefit of continuous multi-factor affirmation of user identity. It uses a client-server model where the clients are these mobile devices listening and reporting perceptible information fingerprints to the server. The mobile devices adopt a simple send and forget model in which it acquires the information and transfers it to the remote server to be further processed. Any association or relationships relevant for this information is stored at server end.

The server portion of Inquest is the web server application which manages client service requests and queries. It preprocesses and stores this information in its database repository associating with the user that claims to use the client mobile device. The worker module of the server application monitors and processes these preprocessed data to resolve useful and applicable information relevant for the Inquest.

Being a feasibility study, the prototype of the Inquest does not address the security concerns typical of a production class client server application. Even though the study looked into ways for effectively storing and managing huge amounts of information relevant to similar kinds of applications, the prototype of the model is not tested against

gigabytes or terabytes of information from a large user base typical of a production class environment.

1.3 USER STORIES

To further explain the Inquest environment and the problem statement it tries to address, the following user stories are presented. The story introduces Bob and Ted who are fictitious users of mobile devices. The devices have necessary connectivity to access the Inquest server web application. These applications of Inquest described in the user stories exist today.

1.3.1 Story 1 – A professional tablet user would like to have a much stringent identify affirmation checks at her device. Tablet has advanced features and applications which she leverages for her work.

Bob buys a new tablet which company X has just introduced to the market. The Tablet offers number of market winning advanced features. Bob found the new features quite useful for his daily work and chose to buy one to complement his work computer on specific applications. Eventually the tablet became an integral part of his professional environment and he uses his tablet more often than his other work place computers.

One day Bob misplaces and leaves his tablet at a cafeteria. Ted on his way into the cafeteria, notices the tablet laying unattended. Eager to help find the owner of the tablet, he grabs it and hands it over to the attendant at register. He hopes that the owner will return to claim the device. The attendant secures the device. Later, the attendant eager to find any clue about owner tries to activate the display by pressing the display activation button. The tablet was either not locked or the attendant was quickly able to work around any existing authentication mechanisms. The attendant opened and glanced through some reports and presentations available from home screen.

1.3.2 Story 2 – A medical doctor accesses patient records and other diagnostic data from smart mobile device. The device is provided by the hospital she works and is often shared within the department. The hospital management would like to have a much stringent identity affirmation checks for the device user.

Ted is a busy and competent medical doctor. As part of their mission to provide their customers with unrelenting and quality care, his hospital has provided him and other relevant staff members with smart devices to access patient records and diagnostic data. Occasionally devices are rotated within the staff. Hospital is aware of risks associated if the device gets into the hands of unintended users.

The hospital would like to have the device monitored non-intrusively for perceptible parameters like device locations, user speech patterns and other user device usage patterns (key presses and swipes) to affirm the identity of the device user.

1.4 CONTRIBUTIONS

Proposed here is a system which can utilize perceptible information from a handheld device to address the challenges implicit in the above mentioned use case scenarios. The Inquest system is a secondary factor method for affirming user identity on a personal handheld computing device. It shall make use of readily available information fingerprints at the personal handheld. Some typical attributes are -

- Device location – Hand held device GPS coordinates reported as latitude and longitude.
- Usage parameters - Device and/or per application usage measures reported at specific intervals.
- Audible speech - Captured and reported in text format in specific intervals during an active device usage.

Behavioral details of the proposed system are documented in detail at next chapter.

The primary contributions contained in this report are,

- A coherent RESTful application programming interface (API) for client to server communication envisioned in the system.
- A prototype of the envisioned system is documented. Design both in macro and micro scale is provided. Assumptions, design considerations and in many instances the rationale of those considerations are documented.
- Understanding of effective database models for similar kind of applications is documented.
- Understanding of feasibility of such a system based on the experimentation with the Inquest prototype is provided.

Reasonable hope is that the content described herein can be further utilized for future work in this area for effective and continuous affirmation of user identity in smart personal mobile devices.

1.5 OUTLINE

Chapter 2 of this report contains the requirements and behavior specification necessary for the envisioned system. Functional and non-functional requirements are captured.

Chapter 3 documents the macro and micro level system design. Various technologies brought together to implement the prototype is described. Understandings from the study on effective database models for the envisioned system is documented. The data model is documented.

Chapter 4 documents the RESTful client-server API interface. HTTP methods and HTTP response status are documented. Example data exchange scenarios are described making use of sequence diagrams.

Chapter 5 documents the results from developing and testing the prototype application. Software engineering metrics are provided. Performance benchmarks from the prototyped environment is captured.

Chapter 6 summarizes the lesson learned from the prototype implementation and outlines the planned future work.

Chapter 2: Requirements and Behavioral Specifications

This chapter describes the functional and non-functional requirements for Inquest. Behavioral specifications are then derived from the mentioned requirements to be considered in the prototype application implementation. Implementation details are covered in the following chapter 3.

2.1 REQUIREMENTS

Requirements for the Inquest application are sourced from the user stories described in chapter 1.

2.1.1 Functional Requirements

Inquest shall implement the following functional requirements to enable the bare minimum features.

Identity Affirmation – The envisioned system shall provide as a method to confirm the identity of the user claiming to be at that personal device within a relaxed period. The term relaxed is mentioned as it can be as prolonged in hours or as short in minutes depending on the specific considerations like availability of information which can be used for the purpose.

Client device ability to monitor and report perceptible information - The service available at the personal mobile device shall collect and process perceptible information readily available on that device like device location, device usage patterns, and audible speech data. The service shall also report this information, unprocessed or preprocessed to a remote server which shall perform necessary analysis tied with a specific user and previously reported data for that user to derive useful identification information.

The device location shall be collected in the form of GPS coordinates [2] reported as a tuple of decimals in the form of <latitude, longitude>. Usage parameters which shall

be captured and reported includes the device and per application usage measures. The audible voice shall be captured and reported in text format at random or implementation specific algorithmically designated intervals. The frequency at which the data is collected or reported to the remote server is not defined and left to be chosen and defined per implementation. In most cases it shall be required that the data shall be captured and reported only when the device is actively used.

Even though the usage of a personal mobile device is most often restricted to a single user, the service shall be considerate of use cases on devices which shall have more than one user and track and associate the captured and reported information to the user actively logged into that device.

Server Administration – Inquest shall expose basic administrative options both at the server and client service environments. Server side options shall include interface for instantiating or connecting to a database, monitoring and reporting usage statistics etc. Client side options shall include interface for tuning or defining the quality of service and specifying the remote cloud based server identification and credential parameters.

Data Representation (Model and Relationships) – Inquest environment shall define the following datatypes and provide the capability to create, delete, update or retrieve them.

- Devices – The mobile devices running the Inquest client service.
- Users – The active users of mobile device subscribed to Inquest service. A user shall use more than one devices.
- Data crumbs – Data feed from Inquest client service reporting the information relevant to a particular user at that device. It includes following attributes.
    - Device location - GPS coordinates reported as a tuple of decimals in the form of <latitude, longitude>.

- Usage parameters - Device usage and per application usage measures reported at specific intervals.
- Audible speech - Captured and reported in text format in specific intervals during an active device usage.

Inquest environment shall provide the mechanism to create and dissolve the associations defined below.

- User uses a particular device.
- Set of known configured users of the particular device.
- A device is reporting specific perceptible information data crumb.

Server Data Persistence – Data objects relevant for a device user shall persist in the Inquest server environment until the user is deleted from that device or unsubscribed from the Inquest service.

Concurrent client Access – Inquest server environment shall gracefully and coherently handle concurrent client access from Inquest client service running at more than one mobile devices.

Data exchange format and interface – Inquest system shall exchange data between client and server environments in a standard data format. The system shall exchange data making use of a standard network protocol to ensure efficient implementation and support from various kind of client devices.

2.1.2 Non-Functional Requirements

Non-functional requirements for Inquest is enlisted below.

Performance – The mobile devices which run the Inquest service shall vary widely in their computing power and network speed. The system shall accept requests and respond within a reasonable time interval defined as 180 seconds. The importance is not given to

efficiency but on the accuracy on affirming the identity of active user claimed on the device within reasonable time window.

Availability – The system shall be available in a continuous manner providing at least 99.99% service availability with near zero down time except during the monthly maintenance window which can last for up to 20 minutes.

Interoperability – The Inquest environment shall support all major personal mobile device operating systems.

Privacy and Security – The Inquest environment shall use network application protocols guarantying higher level of secure transport of information between client and server components. By choosing to subscribe for Inquest service the user agrees that the minimal level of defined perceptible information shall be collected periodically and send to the server component for further processing.

The deployment of Inquest in real world environment shall consider further stringent privacy measures, guaranteeing the collected data is not used for undocumented purposes.

Extensibility – The system shall be designed to consider future expansion of Inquest capabilities to consider more parameters for identity affirmation. These extensions should be possible keeping the external interfaces coherent and unaffected.

Scalability – The system shall be designed to easily scale both vertically (more devices) and horizontally (more users subscribed to Inquest service) to handle increasing user load. The system shall permit millions of records to track millions of users of one or more devices subscribed to Inquest service.

2.2 BEHAVIORAL SPECIFICATIONS

The defined functional and nonfunctional requirements for the Inquest system along with readily available resources for the prototype implementation and current state of main stream computing lead to the definition of following specifications.

- The server system shall be addressable and available over the internet to provide ubiquitous client accessibility upon connected to network.
- The data exchange between the client and server components shall use the JSON (JavaScript Object Notation) notation.
- The client server interface shall be based on HTTP (Hyper Text Transfer Protocol). Client initiates a HTTP request on the Inquest server for any reporting and identity affirmation checks.
- The Inquest service running at the client device shall capture at defined intervals the perceptible information (defined in the form of GPS location, device and application usage information and speech text) and report it to the web present server component associating the information crumb to the user logged into that specific device. The perceptible data crumb shall be only collected on an active user interaction at the device. The collection and reporting shall be periodic in the event of continuous user interaction at the device for efficient and reduced network traffic overhead.
- Identity affirmation checks shall also happen on an active user interaction with the device. Inquest server shall respond to affirmation checks based on its learning potential with a “Positive”, “Negative” or “Don’t Know status. Don’t know status indicate that it is unable to make a coherent prediction based on the available reference vectors. Graceful retries shall be performed by the client application if no response was received within 180 seconds.

- Behavior on a negative identity affirmation from Inquest server is not defined as part of this specification. Client application shall lockout the device or shall prompt for secondary or second factor authentication methods available in that device. It shall log the negative identification event for audit purposes or notify the subscribed user through emails or other alert mechanisms.
- The server component present in the cloud shall be flexible enough in nature to be deployed in one of the many available web container implementations available today.
- The persistent data store at the server shall be only accessed by the server application.
- The cloud based server component shall make use of redundancy to provide a higher level of service availability.

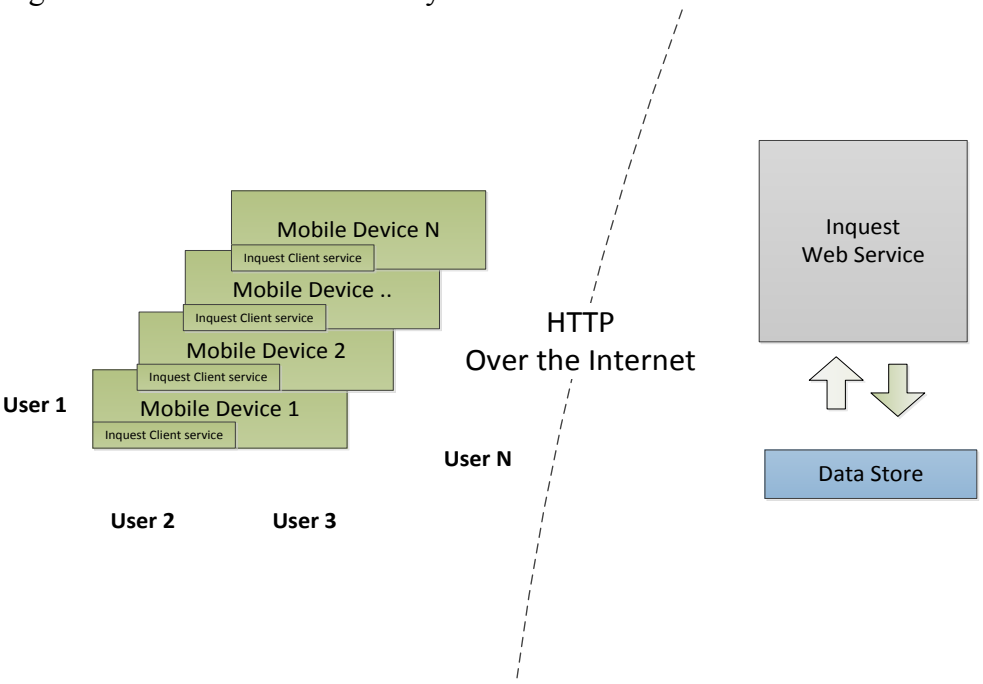

Figure 1: Inquest System Definition

Figure 1 outlines the envisioned system layout. Inquest client service at personal mobile devices shall collect perceptible information finger prints and report to Inquest web service counterpart over the internet. Relevant information throughout this process is persisted at the server component. Inquest server shall respond to affirmation checks based on its learning potential with a “Positive”, “Negative” or “Don’t Know status.

Chapter 3: System Design

This chapter describes the design and component stack in a macro and micro level.

3.1 TECHNOLOGY STACK

Inquest prototype is built up on several free software technologies. Highlighted below are the technologies used at client and server components.

3.1.1 CLIENT SIDE SOFTWARE COMPONENTS

Client side software is developed as an Android application service running on Android lollipop (version 5.xx) based devices. It makes use of Android SDK and google play service APIs to capture perceptible information readily available at the device. Default http client available in Android image is used for performing REST requests at remote server.

The solution assumes that the relevant sensory interfaces are available at the device for the information to be collected by the client application. For example device GPS receiver enables location tracking facility and device audio receiver (mic) enables the ability to capture perceptible speech information.

3.1.2 SERVER SIDE SOFTWARE COMPONENTS

Django – Django is a high level python web framework that encourages rapid web application development and clean pragmatic design [6]. Django implements Model-View-Controller (MVC) architectural pattern. It provides an optional administrative create, read, update and delete interface. Django can run in conjunction with several of the web server environments like Apache, WSGI and Cherokee etc. It officially supports PostgreSQL, MySQL, SQLLite and Orcale while external backend exists to support other

popular database models including NoSQL database models. Django was chosen over other frameworks considering its popularity, richness and ability to enable fast paced development cycles. The number of free and reusable third party packages relevant for the application under consideration is readily available in python environment.

Amazon Web Services (AWS) Elastic Beanstalk – AWS Elastic Beanstalk is a platform as a service (PaaS) that streamlines the setup, deployment and maintenance of web application on Amazon AWS [7]. It is a managed service coupling the servers (Amazon Elastic Compute Cloud (EC2)), database (RDS) and static files (Amazon Simple Storage Service (S3)). AWS Elastic Beanstalk environment was chosen over other available environments to host the server application considering its popularity, feature richness and pricing model.

Pandas Python Data Analysis Library – Pandas [8][9] is an open source BSD licensed Python library providing high performance easy to use data structures and analysis tools for data preprocessing and linear regression.

Scikit-learn library for machine learning in Python – Scikit-learn [10] is an open source, BSD licensed python based library for machine learning applications. The library exposes convenient APIs for variety of data analysis purposes including regression, classifications, clustering, preprocessing, model selection, dimensionality reduction etc.

3.2 INQUEST DESIGN

Inquest system design is captured in figure 2. The Inquest solution adopts a two tier architecture with the client component adopting a fire and forget model which enables a stateless approach at the client. Clients shall not be concerned about the data storage aspects. Data storage responsibility resides with server so that the portability of

client code can be improved. Servers are not concerned about the user interface or end user state, which makes the server implementation simple and scalable.

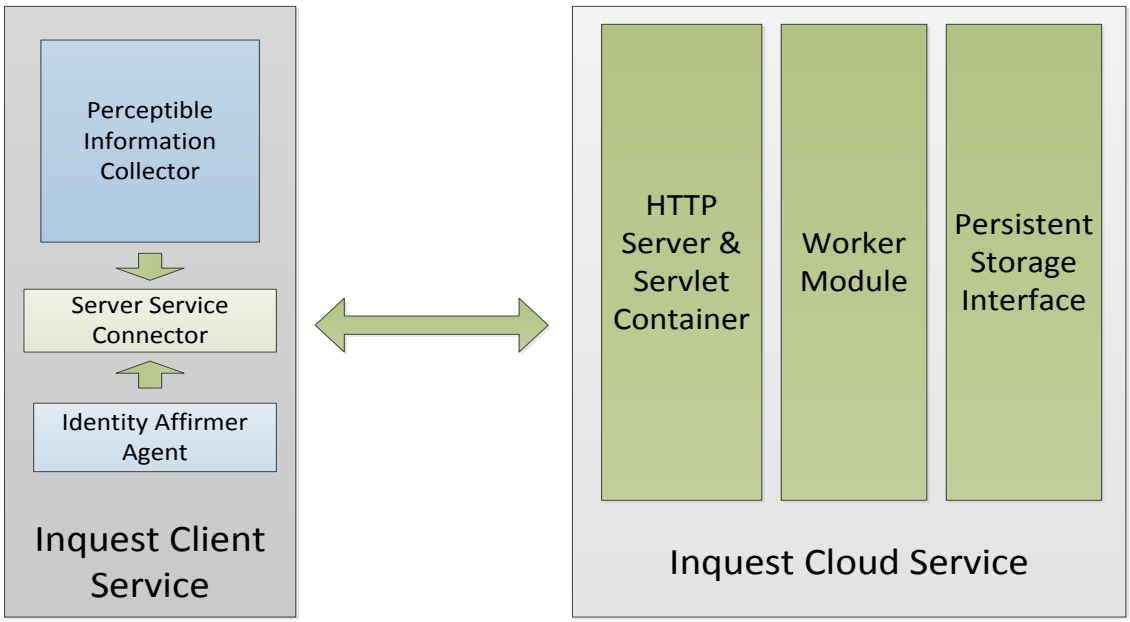

Figure 2: Inquest system architectural components

Perceptible information collector module collects the relevant information at the device and sends the information data crumbs to remote Inquest cloud service through the server service connector module. Connector module abstracts the server connection interface and utilizes REST based HTTP methods for client to server information exchange. Identity affirmer agent queries the server for identity affirmation status following a user action on the device and shall initiate email alerts addressed for the registered user in the event that a negative identity affirmation is flagged.

Inquest cloud service is responsible for storing information reported from the device which it finds relevant for the user identity affirmation process. It accepts crude unprocessed information reported from the client service. The worker module shall

preprocess this information to be transformed into a format relevant for the application use case and persists it for further analysis.

3.3 INQUEST CLIENT SERVICE COMPONENTS

An Android service is an application component that can perform long-running operations in the background and does not provide a user interface. Another application component can start a service and it will continue to run in the background even if the user switches to another application. [14] Secondary components can bind to a service to interact with it and even perform inter process communication (IPC) on that service. A service is "started" when an application component (such as an activity) starts it by calling startService() API. Once started the service can run in the background indefinitely, even if the component that started does not exists anymore.

Inquest client service component architecture is further illustrated in figure 3. Service adopts a fire and forget model. The interval at which the information is collected varies depending on the information being collected. The periods for which the specific device is actively used for last 24 hours is collected once every day by the usage collector module. Location collector module and speech collector module collects the respective information at regular configured intervals during a user activity at that device.

Location collector module at the perceptible information collector makes use of Google Play Service location API to collect the latitude, longitude for that device location and the current timestamp. Speech collector module makes use of Android SDK speech package to capture the voice in digital text format at random intervals during the periods of active device usage. Usage collector module makes use of Android SDK device usage tracking packages to collect the relevant information.

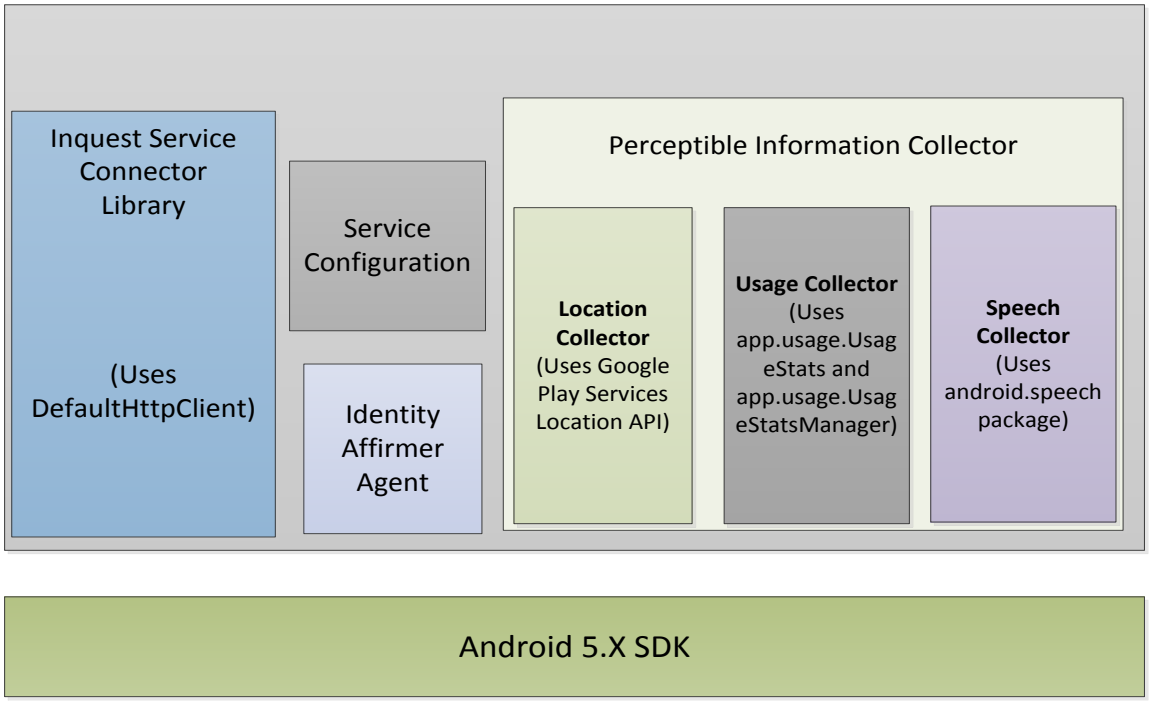

Figure 3: Inquest client components

Service configuration module enables the device user to register for the Inquest service. An email address is accepted as part of the registration to track and identify the registered device user. The user can also configure the quality of service parameter for that client service. A higher value of quality of service parameter settings forces much more frequent data collection and reporting as well as frequent identify affirmation checks comparted to a moderate or lower parameter settings.

3.4 INQUEST CLOUD COMPONENTS

Inquest cloud service exposes a RESTful interface definition which makes use of standard HTTP methods and uses JSON (JavaScript Object Notation) to transfer data

representation between client and server components. Web server application architectural components are captured in figure 4.

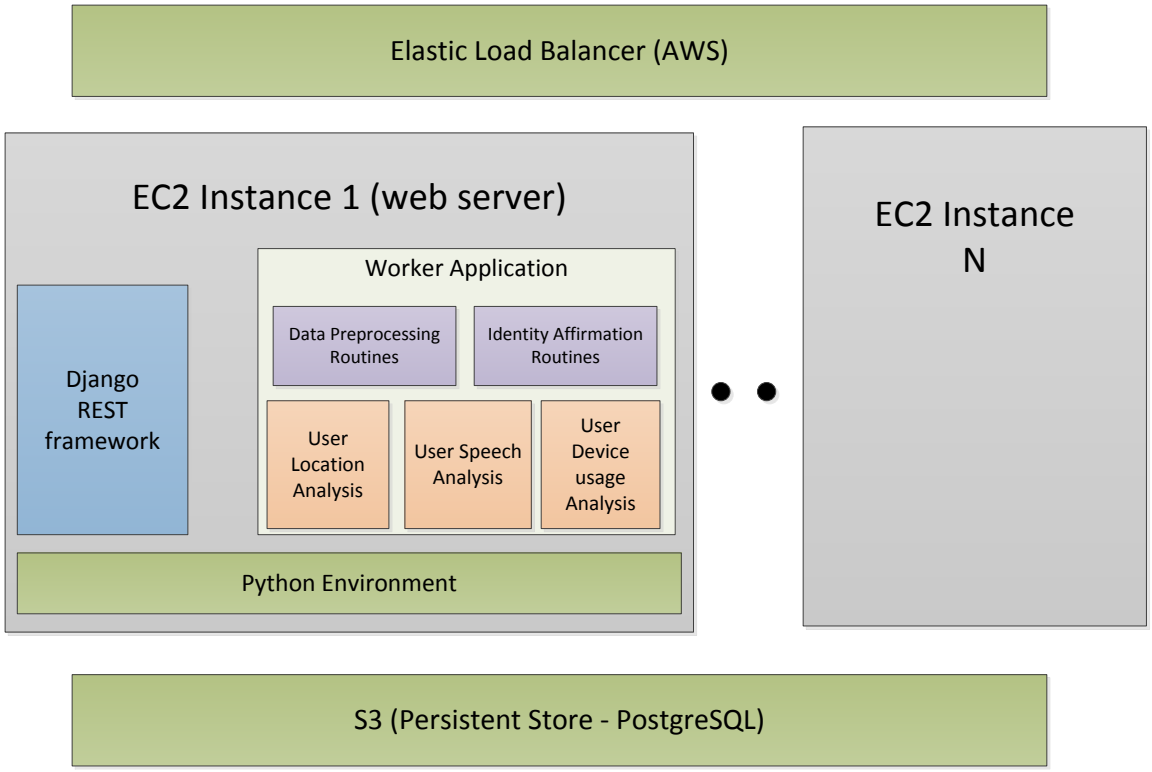

Figure 4: Inquest cloud components

The prototype implementation makes use of Amazon Web Services (AWS) Elastic Beanstalk environment [13] for web server deployment which provides a one layer of abstraction away from the Amazon Elastic Compute Cloud (EC2) layer. To use elastic beanstalk an application is created and uploaded in the form of application source bundle (Python Django package) along with some basic information about the application. Once deployed making use of the AWS management console the Elastic

Beanstalk automatically handles the deployment details of capacity provisioning, load balancing, auto-scaling of active EC2 instances and application health monitoring.

EC2 is the service that allows to create server instances in Amazon cloud which provides enhanced control over the computing resources to the user. The software stack running on the Amazon EC2 instances is dependent on the container type. The container type defines the infrastructural topology and software stack to be used for that environment. Elastic Beanstalk environment with an Apache Tomcat container uses Amazon Linux operating system, Apache webserver and Apache Tomcat software. Elastic load balancer is the Amazon bean stalk component which enables dynamic scaling (dynamic allocation and de-allocation of computing resource) of EC2 instances based on the ongoing service demand and health of existing EC2 instances thus achieving greater levels of application fault tolerance. It distributes the incoming application traffic across the multiple EC2 instances in the cloud.

S3 provides secure, durable and scalable object storage. Amazon S3 offers a range of storage classes designed for different use cases including general purpose storage of frequently accessed data, infrequent access of long-lived less frequently accessed data etc. [15] Amazon S3 is integrated to be functional with variety of database technologies. PostgreSQL is chosen as the database technology for persistent store in the proof of concept implementation considering its readily available support in the AWS environment and scalable nature.

Django rest framework [16] is a powerful toolkit that makes it easy to build web APIs. It is built on top of python and django packages. Django rest framework serializes allow complex data such as query sets and model instances to be converted to native python datatypes that can then be easily rendered into JSON or XML content types, and

also deserialization allowing validated and parsed data to be converted back into complex datatypes. Authentication policies including OAuth1 and OAuth2 are supported.

Inquest server data stores are documented in detail at next section 3.5. Inquest Django Rest framework based web request service module process the incoming web requests and stores the data crumbs in the raw data non persistent table to be picked up by the preprocessing routines. Preprocessing routines parse the data crumbs to separate out the data in a data format which shall be consumed by user location, user speech and user device usage analysis routines. Identity affirmation routines shall run in a periodic manner triggered based on the data updates from device to derive a snap shot of current identity prediction status out of the analysis from the respective routines. Figure 5 represents a model for this processing.

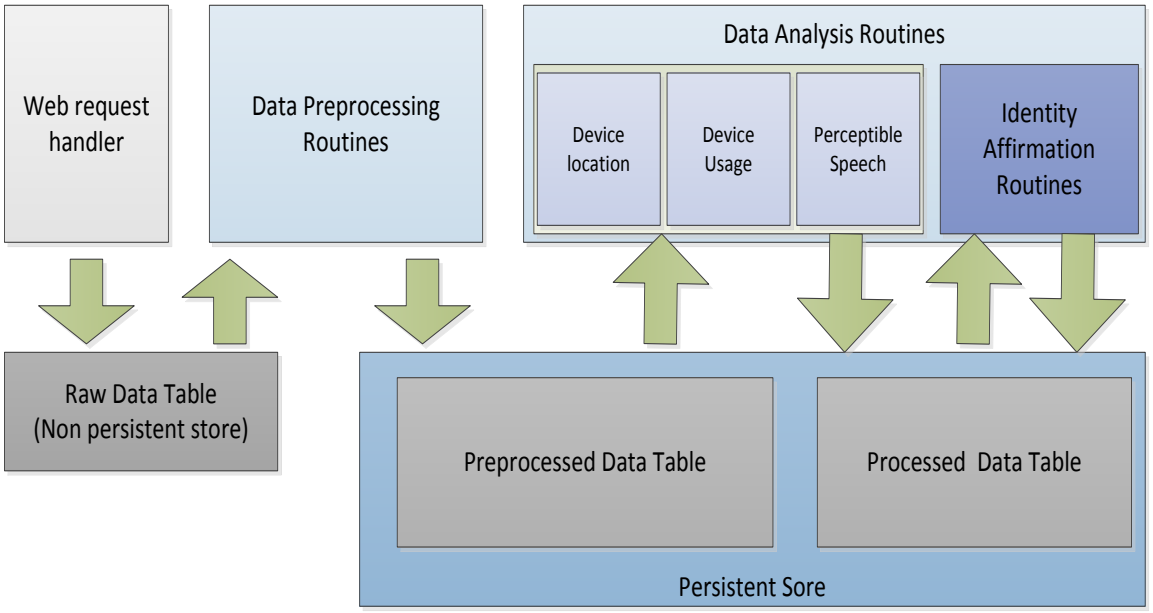

Figure 5: Inquest server data processing model

Data preprocessing routines read raw data from non-persistent raw data table and updates the preprocessed data tables. Device location, device usage and perceptible speech analysis routines makes use of respective tables which contains preprocessed information relevant for those routines. Processed data store which keeps tracks of the unique confidence index values from preprocessed data tables and the identity affirmation index shall be used to derive the current identity affirmation status indices.

3.5 Data Model and Evaluation on Databases Technologies

Inquest server non persistent and persistent data stores are documented in detail in this section. Database technologies evaluated are documented and key learnings are captured.

3.5.1 Non persistent data stores.

Inquest server non persistent data store keeps the raw data crumbs from the devices to be stored temporarily for being picked up by the preprocessing routines. Raw data crumb table attributes are captured below. The following PostgreSQL table is defined in the random access memory.

Device Id
User Id
Data Crumb
Timestamp

Table 1: Temporary in memory raw data crumb table attributes

The data crumbs received from the client device has the following format.

```
{"datacrumb":["location":{"latitude":"<value>","longitude":"<value>","
timestamp":"<value>"},
            "usage":{"timewindow1":"<value>","timewindowN":
"<value>"},
            "speech":{"timestamp":"<value>","text":"<value>"}]}
```

Table 2: JSON encoded device data crumbs

3.5.2 PERSISTENT DATA STORES

Preprocessed information tables and their attributes are captured in this section. Three tables are maintained to capture preprocessed information ready to be consumed by analysis routines one each for storing device location, device usage and perceptible speech information characteristics.

Device location table
Device Id
User Id
Time window zone
Device location latitude normalized
Device location longitude normalized
Frequency
Log frequency
Location confidence index

Table 3: Preprocessed device location information table

Time window zone which is a window to represent a period of duration is used instead of actual timestamps in preprocessed settings. Time window zone is tracked from

1 (12 AM to 12.05 AM period) to 288 (11:55PM to 12AM period) to represent values for every 5 minutes duration.

Location, usage and word confidence indexes are the decimal values indicating the confidence that the device is actually being used by the registered user based on the value of attributes being considered. Confidence index values varies between 0 and 1, with higher values representing a higher confidence. The frequency represents the number of times the device is used cumulatively by that user during the specific time window zone. Further details on preprocessed data attributes are documented in section 3.6.

Device usage stats table
User Id
Time window zone
Frequency
Log Frequency used
Usage confidence index

Table 4: Preprocessed device usage statistics table

A keyword in a perceptible speech information table shall represent a combination of one of more words. Three most significant words are identified from a data crumb update and tracked as primary, secondary and tertiary keyword. Keyword frequency represents the number of times the specific keyword is referenced during the time window zone. The raw frequency value is transformed to log values to dampen the possibility of extra importance given to higher values on subsequent computations.

Perceptible speech Information
User Id
Primary keyword
Secondary keyword
Tertiary keyword
Frequency primary
Log frequency primary
Frequency secondary
Log frequency secondary
Frequency tertiary
Log frequency tertiary
Time window zone
Word confidence index

Table 5: Preprocessed perceptible speech information table

Device Id
User Id
Timestamp
Time window zone
Location confidence index
Usage confidence index
Word confidence index
Identity affirmation index

Table 6: Confidence index table

Processed data table attributes are captured as follows. This table is updated by worker processes when a unique combination of confidence index values are identified from the preprocessed data tables to derive the current identity affirmation status.

3.5.3 EVALUATION OF DATABASE TECHNOLOGIES

This section looks into various database technologies which were looked into and studied upon before choosing the data store implementation for proof of concept. Redis based in memory store was identified as a good choice for storing raw data crumb table due to its fast and simple key value based data access ability. A NoSQL based document store style database was found ideal for storing preprocessed and processed information tables, with documents modeled to track information about each registered user.

But in the context of proof of concept, Amazon S3 PostgreSQL database was chosen to be used for storing both in memory and persistent data considering its inexpensive and readily available support in AWS environments. It is expected that the specific representation shall be functioning reasonably well in the limited user proof of concept environments.

3.5.3.1 RELATIONAL DATABASE TECHNOLOGIES

A relational database technology models data using a structure and predicate consistent with first order predicate logic (primary key), where all data is represented in terms of tuples, which are grouped into relations. Most relational databases uses Structured Query Language (SQL) based data definition and query methods. PostgreSQL was looked into as convenient choice for relational database based implementation considering its adoption rate in scaled environments and popular integration in AWS S3 environment.

PostgreSQL is a cross platform object relational database management system where the objects, classes and inheritance are directly supported in database schemas and query language. PostgreSQL is ACID compliant and transactional. It is known to handle workloads ranging from small single processor application environments to large scale internet facing applications with large concurrent user base.

3.5.3.2 NON-RELATIONAL (NO SQL) DATABASE TECHNOLOGIES

A NoSQL or non-relational database provides a mechanism for storage and retrieval of data that is modeled in means other than the tabular relations used in relational databases. NoSQL non-relational database has gained a wider adoption and is popular in large scale user facing internet applications. In this section popular database technologies representing various types of non-relational schema layouts are discussed. Specifically popular representatives of graph based, document based and key value based data store technologies are discussed.

3.5.3.2.1 GRAPH BASED DATABASES – NEO4J

Graph based databases are best used for graph style rich or complex interconnected data. Some ideal use cases are for searching routes in social relations, road or transport maps and other network topologies. Like other database management systems it makes use of Create, Read, Update and Delete (CRUD) methods but exposes a graph based data model. Neo4j uses native graph storage that is optimized and designed for storing and managing graphs and claims to leverage index-free adjacency [17].

The most compelling reason for choosing a graph database could be claimed as the performance gains when dealing with interconnected connected large scale data versus when handled on a relational databases or other NOSQL stores.

3.5.3.2.2 DOCUMENT STORE – MONGODB

A cross platform no SQL document oriented database, MongoDB uses JSON like documents with dynamic schemas for data Storage. [18] Data in MongoDB has a flexible schema. Unlike relational databases, where you must determine and declare a table schema before adding content into it, MongoDB's collections do not enforce document structure. Queries are javascript expressions.

MongoDB retains some friendly properties of SQL like query and indexing. The concept of references shall be used to define normalized data models. It is possible to embed related data in a single structure or document to form rich documents. (Such schema is generally known as de-normalized models).

MapReduce framework can be used for batch processing of data and aggregation operations. MongoDB supports high availability with replica sets which consists of two or more copies of data. It supports readily integration with number of languages including python which is the choice of language for Inquest server side development. It supports integrated text search which is handy for some of the Inquest server side use cases.

3.5.3.2.3 KEY VALUE & DOCUMENT STYLE STORE – DYNAMODB

DynamoDB is a proprietary no SQL database that supports both document and key-value style store models. [19] Amazon who is the provider for this database technology brings in readily integration and flexibility advantage to it by tying it with its popular AWS framework. It is integrated with AWS lambda to provide triggers which enables automatic reaction to data changes at application requirement. AWS Identity and access management integration allows for fine grained user access control on data access.

The concept of global secondary index associates each table item in the DynamoDB table to have a hash key and an optional range key allowing fast query lookups. Amazon elastic map reduce managed Hadoop service allows data store

scalability for data intensive work flows. Integration with Mahout's mature map reduce algorithms allows access to scalable and performance intensive data analysis and machine learning algorithms.

3.5.3.2.4 KEY VALUE STORE – REDIS

Redis is primarily an in memory data structure store often used as a database, cache and message broker. [20] Redis supports built-in replication, different levels of on disk persistence, high availability via Redis Sentinel [21] (providing capability for real time monitoring, and automatic failover) and automatic partitioning with Redis cluster.

Redis is not a plain key-value store, but acts as a data structure server supporting various simple and complex data structures, where string keys can be used to reference complex data structure values. Example data structures supported by the Redis are linked lists (collection of strings), sets (collection of unique unsorted strings), sorted sets, hash maps, bit maps etc.

Redis can be best used for rapidly changing data of foreseeable size. Some simple examples are real time communication interface, leader board etc.

3.6 METHODS FOR IDENTITY AFFIRMATION

This section discusses the preprocessing and data analysis techniques employed to derive the identity affirmation status for a registered and claimed user at that device. Effectiveness of various algorithms attempted is documented in chapter 5.

3.6.1 PREPROCESSING TECHNIQUES

Data crumbs reported from devices stored at temporary in memory tables by the HTTP request handlers shall be consumed by the preprocessing routines for further digest to a format consumable by analytics routines. Preprocessed persistent table content values are derived by respective preprocessing routines.

Frequency is represented *function (frequency)* format in the preprocessed settings. The raw frequency generally reflect on the importance or salient nature of that specific attribute. The higher the frequency, it better represents its importance. But as this value gets higher, it could generally influence the results of further computations and hence log transformation is used to dampen the raw frequency values.

Function (frequency) = 1 + log (frequency), for frequency value greater than 0, while it is left as zero for zero values.

3.6.1.1 DIGESTING DEVICE LOCATION INFORMATION CRUMBS

Device location table described in section 3.5.2 is the data store table used to store the preprocessed device location attributes. Location attribute from the raw data crumb in-memory table is used to derive the information. Time window zone ID which represents a period of duration is used instead of actual timestamps in preprocessed settings. Log of frequency is used by the analytics algorithms instead of raw frequency values. Latitude and longitude values are normalized to point to a prior documented value, if the newly reported location coordinates are within 100 feet from the prior documented value. Location confidence index is generated by the algorithm as described in the section 3.6.2.2.

3.6.1.2 DIGESTING DEVICE USAGE INFORMATION CRUMBS

Device usage stats table described in section 3.5.2 is the data store table used to store the preprocessed device usage attributes. Usage attribute from the raw data crumb in-memory table is used to derive the information. Time window zone ID which represents a period of duration is used instead of actual timestamps in preprocessed settings. Log of frequency is used by the analytics algorithms instead of raw frequency

values. Usage confidence index is generated by the algorithm as described in the section 3.6.2.2.

3.6.1.3 DIGESTING DEVICE PERCEPTIBLE SPEECH INFORMATION CRUMBS

Perceptible speech information table described in section 3.5.2 is the data store table used to store the preprocessed perceptible speech attributes. Speech attribute from the raw data crumb in-memory table is used to derive the information. From the group of words reported the three most prominent words are chosen to be included in this table. Further details on preprocessing techniques on text content is documented in next section 3.6.1.3.1. Time window zone ID which represents a period of duration is used instead of actual timestamps in preprocessed settings. Log of frequency is used by the analytics algorithms instead of raw frequency values. Word confidence index is generated by the algorithm as described in the section 3.6.2.2.

3.6.1.3.1 PREPROCESSING TECHNIQUES ON TEXT CONTENT

Unstructured text is a very common form of data and is of specific interest in Inquest application context considering that the perceptible speech collected at the device at specific intervals shall be available in text format and shall be utilized at the server to derive meaningful information. In most general terms, text mining will turn text into numbers or meaningful indices, which can be then incorporated into supervised or unsupervised predictive analysis techniques to derive meaningful information.

The incoming English based text data is preprocessed to exclude common stop words [11]. Different grammatical forms of same word is combined. Then in the simplest level words found shall be indexed and counted to derive a table of words or a matrix of frequencies that tracks the number of times the word was referenced during the specific time window. Another important preprocessing technique is the stemming of the words

which try to reduce the words to their roots so that different grammatical forms or declinations of verbs are identified and indexed as same word. Text is converted into consistent lower or upper case representation. Another complex slightly more involved technique is combining words or phrases which are synonyms denoting unique meanings.

Once this table of unique words is derived standard statistical and data mining techniques shall be applied to derive dimensions or clusters indicating useful information. In the specific context of Inquest server use case, three most prominent words from every device information update is used.

3.6.2 ANALYSIS METHODS

This section describes the predictive models which were tried out and provides an outlook on effectiveness of each approaches in the context of the defined data model for preprocessed attributes. Simulated test and training data sets for much of the analysis were derived as documented in following section.

3.6.2.1 TRAINING AND TEST SAMPLES

The training and test data used for evaluation of methods and data model was simulated from public social media posts of selected users. Text from public posts is used to fill in speech attribute, approximate GPS coordinates for the identified location is used to fill in location attribute and identifiable usage time stamps are used to fill in usage attribute in data crumbs for gathering training samples. Similarly a bunch of related and unrelated posts for same set of users and a whole new set us users are used to gather test data samples. It is assumed that this information will closely match the kind of information gathered in real time Inquest environment and shall be used as the training dataset until a much more reliable training dataset can be generated from a production like environment.

About 50 each sample set from 20 users are collected as training set for model evaluation.

3.6.2.2 PREDICTIVE MODELS

This section briefly describes the models experimented in deriving the location, usage, and word confidence index and identity affirmation index values. Information on the respective models effectiveness from the Inquest data set is documented in chapter 5.

3.6.2.2.1 DECISION TREE BASED MODEL

In a decision tree based model each tree leaf node gets assigned a class label which will be either a 0 or 1 with the data set being considered. The non-terminal nodes, which includes the root and other intermediate nodes contains the selected attribute or feature test conditions to separate records that have different characteristics. [24] Once the decision tree model is constructed based on the training data set, providing a classification for test records is straightforward. Starting from the root node the prediction applies the test condition to the record and follow the appropriate branch based on the outcome of the test, which will lead to another internal node to which the test condition is further applied or to a leaf node. Once the leaf node is reached, the class label associated with that leaf node is assigned to the sample as the predicted target value.

Scikit learn DecisionTreeClassifier is used for evaluation. Predit_proba capability of the classifier is used to derive the confidence index values. [24]

3.6.2.2.2 RANDOM FOREST BASED MODEL

A random forest is a classifier consisting of a collection of tree structured classifiers $\{h(x,\Theta k), k=1, \ldots N\}$ where the $\{\Theta k\}$ are independent identically distributed random vectors and each tree casts a unit vote for the most popular class at input x [25]. Each tree in the ensemble is built from the sample drawn from the training set. When a

node is split during the construction of the tree, the split chosen is often not the best split among the features, but is the best split among a random subset of features. Randomly m predictors is chosen as split candidates from the full set of p predictors and split is allowed to use only one of those m predictors. A fresh sample of m predictors is taken at each split where m is approximately equal to the square root of the total number of available predictor p. Due to this induced randomness even though bias of the resulting model is slightly increased in the process, due to averaging the results from trees in the forest to generate the model prediction for a sample, the variance decreases usually more to compensate for the increase in bias resulting in overall stronger model.

Scikit learn ensemble RandomForestClassifier in default Gini impurity criterion is used for evaluation. Predit_proba capability of the classifier is used to derive the confidence index values.

3.6.2.2.3 GRADIENT BOOSTING MODEL

Gradient boosting machines are ensemble method that uses weaker models like decision trees to create an accurate prediction. It can be used for regression and classification. Gradient boosting accepts parameters including number of trees, tree depth (Number of tree splits allowed), regularization and shrinkage (Over fitting can happen when optimizing the model too closely to the training set. Shrinkage allows regularization to control over fitting. A learning rate parameter is used that takes a value from $0 < v <= 1$, where 1 indicates no regularization) to create a trained model [26].

Scikit learn ensemble GradientBoostingClassifier in default deviance loss function is used for evaluation. Predit_proba capability of the classifier is used to derive the confidence index values.

3.6.2.2.4 SUPPORT VECTOR REGRESSION MODEL

Support vector machines (SVMs) are a set of supervised learning methods used for classification and regression problems. This model was considered for evaluation considering its effectiveness in text analysis and in high dimensional data model spaces, specifically its usefulness in cases where number of dimensions is greater than the number of samples.

It uses a subset of training points in the decision functions which makes it memory efficient. LinearSVR [27] method of support vector regression was employed on the data set to derive the probabilistic estimates of the confidence index values and identity affirmation index values. SVR model depends only on a subset of the training data as the cost function for the model ignores any training data close to the model prediction. From a general perspective SVMs are helpful in text and hypertext categorization as their application can significantly reduce the need for labeled training instances in both the standard inductive and trans-inductive settings.

3.6.2.2.5 NAIVE BAYES MODEL

Naïve Bayes based models are a category of supervised learning methods which are probabilistic classifiers based on applying Bayes theorem with strong independence assumptions between the features. It was initially adopted for text retrieval methods and remains a popular baseline method for text categorization. Combined with appropriate preprocessing techniques it can be further competitive against advanced methods like support vector machines.

Scikit learn MultinomialNB [28] classifier model was used to evaluation the word confidence index values. Incremental fit model was tried out in addition to fit based Naïve Bayes classifiers.

3.6.2.2.6 LINEAR REGRESSION

Ordinary least squares linear regression with fit intercept is used. Linear_model.LinearRegression scikit learn function is used to predict.

3.6.3 EVALUATION CONTROL FLOW

Flow of control at the server application on a new data crumb update at the temporary table is depicted in figure 7.

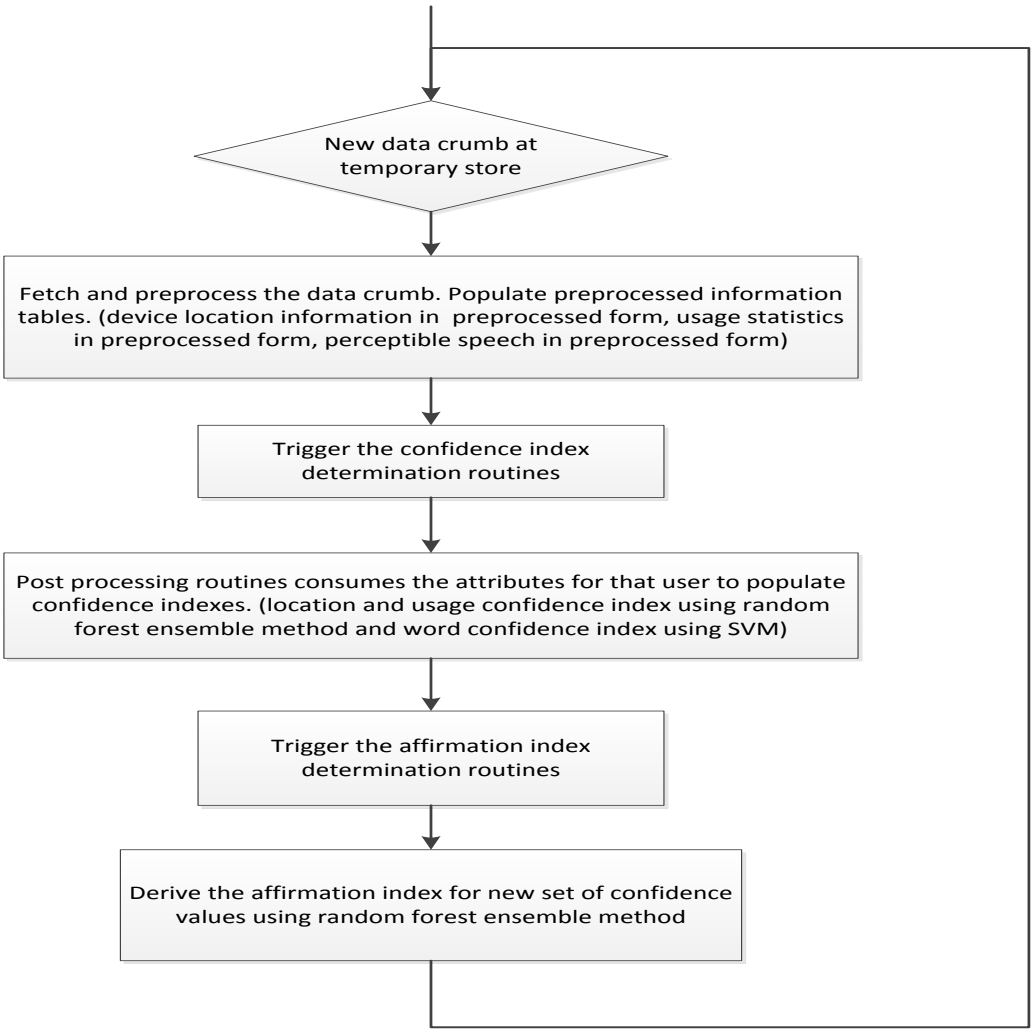

Figure 6: Server application evaluation control flow

A row of attributes for each of the preprocessed tables are populated making use of attributes derived from a data crumb updates. Once the confidence index values are derived from the information based on that atomic device update, affirmation index is identified. When a device queries for the current affirmation status server shall consider the identity affirmation indices for past N hours and return with a normalized user identify affirmation prediction which reflects its understanding for that N hours. N is derived out of the quality of service registration settings for the device user. For a comparatively aggressive identity affirmation check subscription from user, N shall be configured for lower values. Inquest client service is programmed for frequent data crumb updates and checks in the event of a negative trending affirmation indices.

Chapter 4: RESTful Client Server Interface and APIs

Inquest web server interface is modeled as RESTful. This chapter is dedicated to the interface definition to highlight the influence the REST concepts had on the overall design.

Representational State Transfer (REST) is a software architectural style for web services which was initially proposed by Roy Thomas Fielding in his PHD dissertation "Architectural Styles and Design of Network-based Software Architectures" [3]. REST proposes a coordinated set of 6 constraints to the design of components in a networked hypermedia subsystem. The only optional constraint of REST architecture is code on demand. If a service violates any other constraint, it cannot be strictly referred as RESTful. The REST constraints are listed below to set a background for the considerations in defining Inquest request and response structure.

1) Uniform Interface – Uniform interface constraint defines the interface between client and servers. It simplifies and decouples the architecture enabling each part to evolve independently. The interface definition has following guiding principles. a) Individual resources are identified in requests using URIs as resource identifiers. b) When client holds a representation of resource, it has enough information to modify or delete the resource on the server, provided it has the necessary permissions. c) Request and response messages are self-descriptive so that each message includes enough information to describe how to process that message. d) Hypermedia (transcript of information exchange) is defined to represent the state. Clients deliver state via body contents, query string parameters, request headers and

the requested URI (the resource names). Services deliver state to the clients through body content, response codes and response headers.

2) Stateless – The necessary state to handle the request is contained within the request itself as part of the URI, query string parameters, body or headers. Web server containers shall not be required to keep track of the user sessions. The URI uniquely identifies the resource and the body contains the state or action required on that resource. The server once it completes its processing the information that is relevant to the client about its state is communicated back to the client through its response headers and body content.
3) Cacheable - Web clients often caches responses for server components. Well managed caching partially or completely eliminates some client server interactions, thus improving performance and scalability. Server responses must therefore, implicitly or explicitly define themselves as cacheable or not to prevent clients using inappropriate or stale data.
4) Client-Server separation – Clients are not concerned about the data storage. Data storage responsibility resides with server so that the portability of client code can be improved. Servers are not concerned about the user interface or end user state, which makes the server implementation simple and scalable.
5) Layered server implementation – Client cannot realize if it is connected directly to the end server or to some intermediary. Intermediary servers enable load balancing and thus improves scalability.
6) Code on Demand (optional) – Servers are able to temporarily extend or customize the functionality of a client by transferring logic to the client, so that the client can execute it. Java applets or client side scripts such as JavaScript are example implementation methods.

As Fielding highlighted in his dissertation REST emphasizes scalability of component interactions, generality of interfaces and independent deployment of components. The REST service uses standard HTTP (Hyper Text Transfer Protocol) methods and uses JSON (JavaScript Object Notation) to transfer object representation. It is recommended that a reader who is not familiar with REST shall read [3].

4.1 HTTP REQUESTS

Inquest client service uses following standard HTTP methods for all interactions with its server counterpart. Inquest makes use of the standard CRUD (Create, Retrieve, Update and Delete) operations for all persisted information objects.

4.1.1 HTTP GET

HTTP GET method is used for all information retrieval activities like retrieving the current identity affirmation status.

4.1.2 HTTP POST

HTTP POST method is used for all create operations. Typical usage scenario is on a user enrollment for Inquest service at a device. The user enrollment shall be a new user enrolling his device for Inquest of an existing Inquest service user enrolling an additional device for the service.

4.1.3 HTTP PUT

HTTP PUT method is used for all update operations. PUT method is used for perceptible data crumb updates for that user claimed to be logged into that device.

4.1.4 HTTP DELETE

HTTP DELETE method is used for unsubscribing a user from Inquest service at a specific device.

4.2 HTTP RESPONSES

This section documents the Inquest server responses to client service requests. RFC 2616 [5] discusses in details the various HTTP response codes and its purpose. JSON is used for transferring object representation as part of response body.

4.2.1 SUCCESSFUL RESPONSE CODES (2XX)

This section highlight the response codes used in Inquest solution for server component to indicate successful completion of a client service request.

Ok (200) – Represents the request succeeded. The information returned in the response is dependent on the method used in the request. For example information returned in a GET request is a representation of the resource requested.

Created (201) – Represents that the request was fulfilled and resulted in a new resource being created. The newly created resource can be referenced from the URI returned as part of the response entity. The response includes an entity containing a list of resource characteristics and location from which the user agent can find the most relevant information.

No Content (204) – Represent that the server has fulfilled the request and does not need to return any entity object representation. The client shall not change its document view from that which caused the request to be sent. An example situation which Inquest returns no content response is when perceptible information is updated for the claimed user of that device or when a user is unsubscribed from Inquest service successfully for a specific device in response to the client service request.

4.2.2 CLIENT ERROR RESPONSE CODES (4XX)

This section highlight the response codes used in Inquest solution for server component to indicate an error in the client service request structure.

Bad Request (401) – Represents the request could not be understood at server. It is expected that the client shall not repeat the request without modifications.

Not found (404) – Represents the server was not able to find a matching request URI. Inquest returns this status if the client request references an unknown user for that device.

Request Entity Too Large (413) – Represents the server is not able to process the request as the request entity is larger than the server implementation is able to process. Inquest server shall accept and process HTTP requests with a maximum of 1K request content size. If the client service is required to send information at once which is more than 1K it is expected that client sends it as more than one request.

Request URI too long (414) – Represents the server is not able to interpret the request URI. A rare scenario this error shall be returned is when a client has inappropriately converted a POST or PUT request to GET request with long query information.

Unsupported Media Type (415) – Represents the server is unable to process the requested content format. Inquest server expect that the service request is encoded JSON encoded.

4.2.3 SERVER INTERNAL ERROR RESPONSE CODES (5XX)

This section highlight the response codes used in Inquest solution for server component to indicate an error during the processing of a client service request.

Internal Server Error (500) – Represents that the server encountered an unexpected condition which prevented it from processing the request.

Not Implemented (501) – Represents that the server does not support the functionality requested. This is returned when the server does not recognize the request method.

Service Unavailable (503) – Represents that the server is unable to process the request due to temporary overloading or server maintenance. The implication is that the temporary condition shall be alleviated after some delay.

4.3 URIs

The uniform resource identifier (URI) is a string of characters used to identify the name of a computing resource enabling interactions with the representation of that resource over the network. For easy understanding comparison shall be made with Uniform resource locator (URL) which is referred informally as a web address which is the most common form of URI.

All Inquest server requests shall be HTTP requests initiated on a URI. This sections reviews Inquest URI layout. Inquest URIs are defined relative to the base URI defined as http://<hostname>:<port>/. The base URI structure is dependent on the servlet container in use for deploying the web service and specifics shall vary based on the Inquest server deployment.

Operation on Resource	**HTTP request type**	**URI**
Subscribing a user for Inquest service at a device	POST	<base URI>/subscribe/user/<userid>/device/<deviceid>
Unsubscribing an Inquest user from a device	DELETE	<base URI>/unsubscribe/user/<userid>/device/<deviceid>
Update perceptible information from a device	PUT	<base URI>/datacrumb/user/<userid>/device/<deviceid>
Retrieve identity affirmation status	GET	<base URI>/identitystatus/user/<userid>/device/<deviceid>

Table 7: Inquest resource URIs

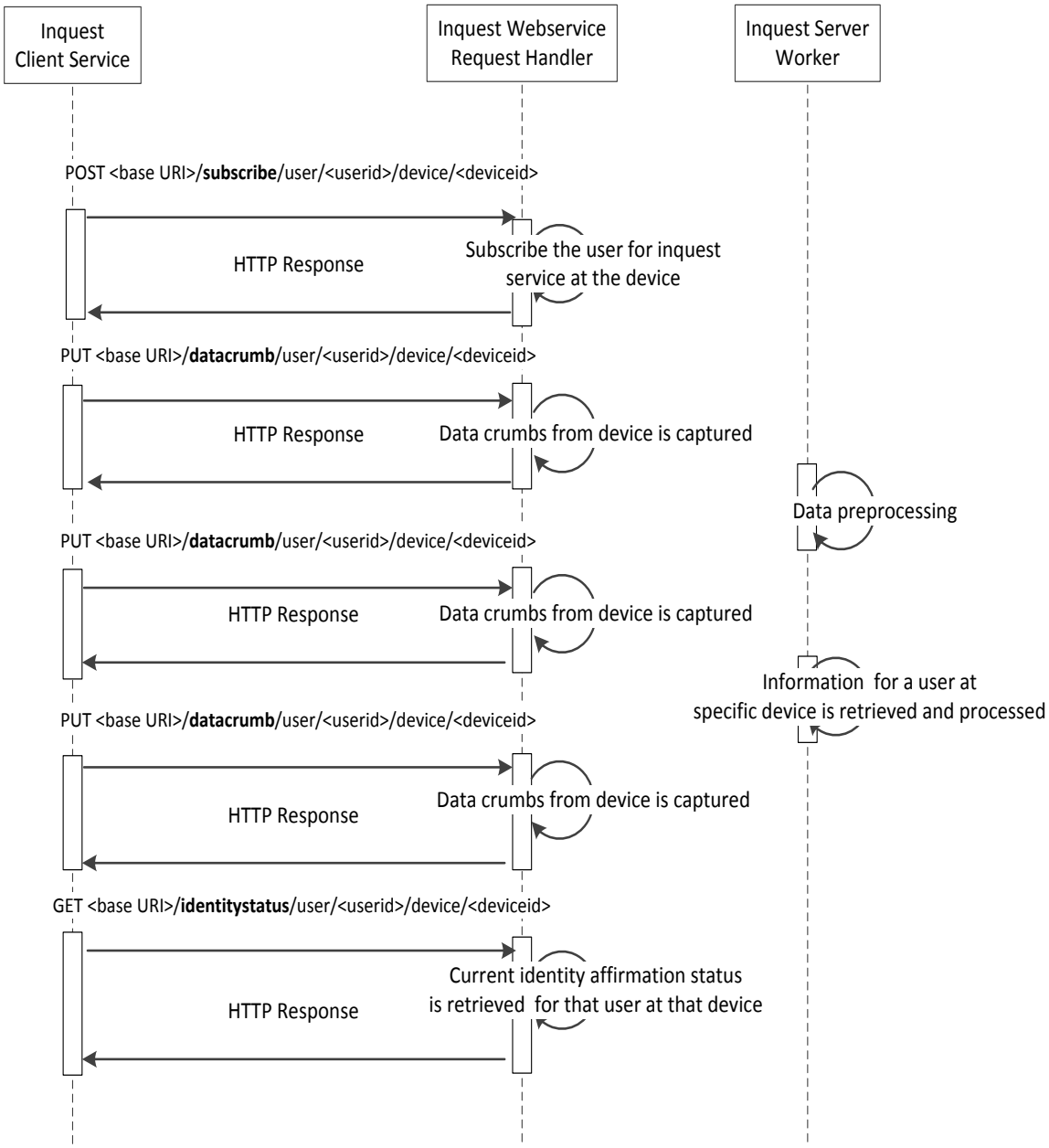

Figure 7: Inquest client server API request response flow

API interface is carefully designed to ensure that the URIs are self-expressive and meaningful. The attempted operation can be easily described by reading the URI combined with the HTTP request type. Table 1 enlists the Inquest operations and the URI structure used to address the respective resource under consideration.

URIs are relative to the system base URI. <userid> represents the identification of the user claimed at that device. <deviceid> represents the unique identifier for that device, which shall be retrieved from device making use of the convenient APIs exposed by the device operating system SDK.

Figure 7 captures an example snapshot of client server request response sequence flow. Once client service subscribes for the Inquest service with the server, it shall collect the relevant data crumbs and reports the information at the server making use of *datacrumb* PUT HTTP interface. Identity affirmation status shall be queried making use of *identitystatus* GET HTTP request, which shall respond back with the identity affirmation status for the active user at that device.

Inquest server worker shall preprocess the incoming data crumbs and shall device meaningful information out of it. Figure 8 captures an example snapshot of client device service subscription and un-subscription flow.

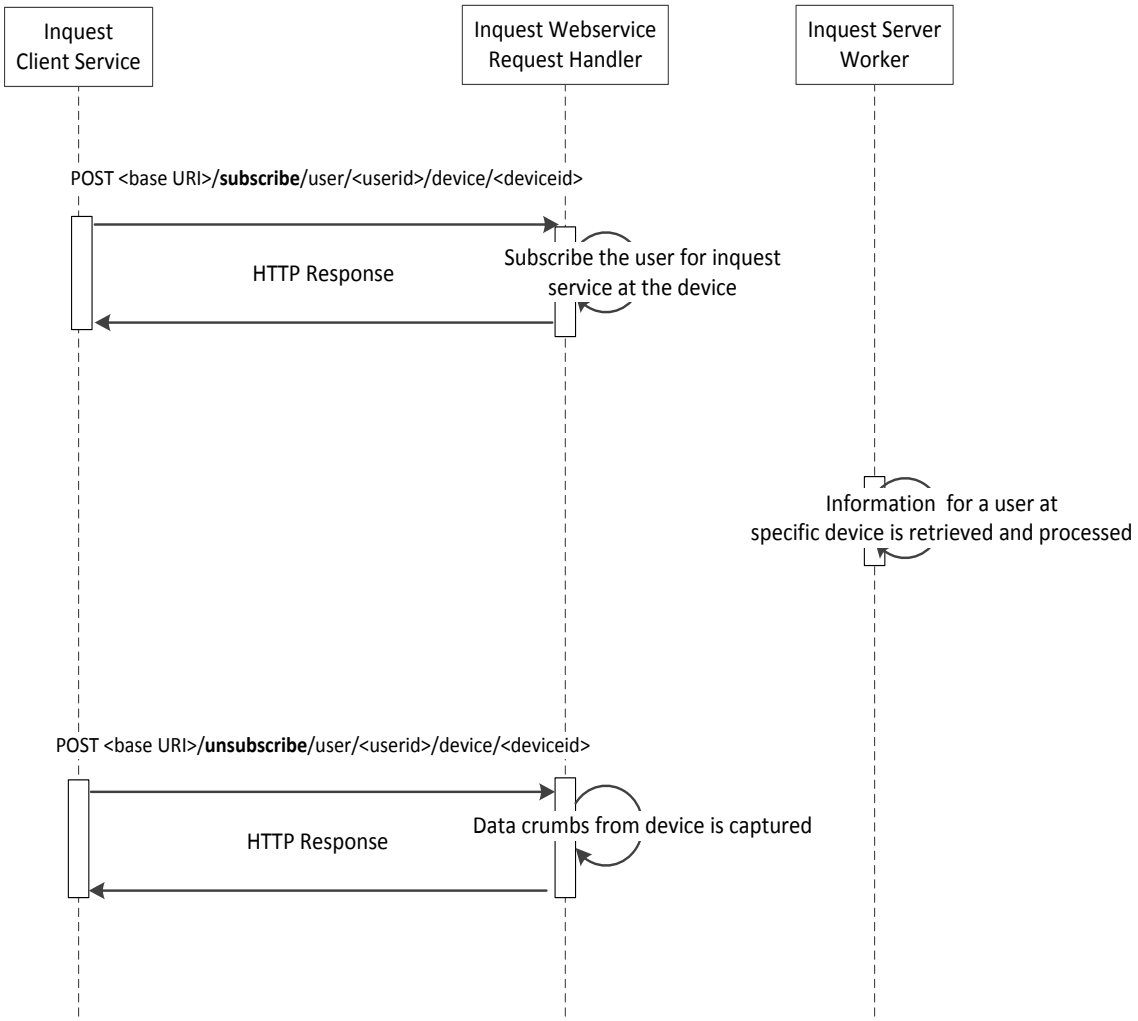

Figure 8: Client Inquest subscription and un-subscription flow

Chapter 5: Results

Even though Inquest is a proof of concept with many critical code paths expected to be left un-optimized and untested in scaled environment, benchmarking is performed to evaluate the feasibility of the approach and system. Attempt is made to capture the results available at the time of writing this report and some of the software engineering metrics measures.

5.1 TEST ENVIRONMENT

A two steps approach has been carried out, of which the first is to evaluate the analysis methods against the data models so that a model can be chosen for further consideration.

The training and test data used for evaluation of methods and data model was simulated from public social media posts of selected users. Text from public posts is used to fill in speech attribute, approximate GPS coordinates for the identified location is used to fill in location attribute and identifiable usage time stamps are used to fill in usage attribute in data crumbs for gathering training samples. Similarly a bunch of related and unrelated posts for same set of users and a whole new set us users are used to gather test data samples. It is assumed that this information will closely match the kind of information gathered in real time Inquest environment and shall be used as the training dataset until a much more reliable training dataset can be generated from a real device user environment.

The second step is to evaluate the system from an end to end use case execution environment. Tests so far have been mostly on a single device environment. Simulated training data set is used which is expected to be gradually replaced by the real data once enough of those data set is available.

5.2 EVALUATION OF THE ANALYSIS METHODS

Methods evaluated for deriving the post processed table values and affirmation indices are presented. Method's effectiveness based on cross validation against the simulated training and test dataset is documented. Cross_validation.cross_val_score is used for determining the cross validation score. Scikit learn metrics.accuracy_score is used to derive the accuracy score.

5.2.1 LOCATION CONFIDENCE INDEX

Location confidence index is filled in as a value of predicted probability. The value is populated at the device location table alongside the attributes used to derive the value. An entry is also made at the confidence index table which will be used for deriving affirmation index. The following listing provides a comparison of model performance based on cross validation and accuracy score. Random forest ensemble is chosen to be used for deriving the value at server application.

Evaluation method	**Cross validation score**	**Accuracy score**
Linear regression	0.91146	0.890000
Decision tree	0.92386	0.911423
Random forest ensemble	0.93897	0.918424
Gradient boosting ensemble	0.93189	0.901487

Table 8: Model evaluation for deriving location confidence index

5.2.2 USAGE CONFIDENCE INDEX

Usage confidence index is filled in as a value of predicted probability at the device usage table alongside the attributes used to derive the value. An entry is also made at the confidence index table which will be used for deriving affirmation index. The following listing provides a comparison of model performance based on cross validation

and accuracy score. Random forest ensemble is chosen to be used for deriving the value at server application.

Evaluation method	**Cross validation score**	**Accuracy score**
Linear regression (least squares)	0.84672	0.830000
Decision tree	0.84732	0.839211
Random forest ensemble	0.84943	0.841455
Gradient boosting ensemble	0.84740	0.839901

Table 9: Model evaluation for deriving usage confidence index

5.2.3 WORD CONFIDENCE INDEX

Evaluation method	**Cross validation score**	**Accuracy score**
Support vector regression	0.76002	0.754147
Support vector machine with RBF kernel	0.81791	0.801733
Naïve Bayes	0.80732	0.799211

Table 10: Model evaluation for deriving word confidence index

Word confidence index is filled in as a value of predicted probability at the perceptible speech information table alongside the attributes used to derive the value. An entry is also made at the confidence index table which will be used for deriving affirmation index alongside the device and usage confidence index values derived from the data crumb. The following listing provides a comparison of model performance based on cross validation and accuracy score. Support vector machine is chosen to be used for deriving the value at server application.

5.2.4 IDENTITY AFFIRMATION INDEX

Identity affirmation index is filled in as a value of predicted probability alongside the attributes (device location, usage and word confidence index values derived from a specific device data crumb update) used to derive the value. The following listing provides a comparison of model performance based on cross validation and accuracy score. Random forest ensemble is chosen to be used for deriving affirmation index at server application.

Evaluation method	**Cross validation score**	**Accuracy score**
Linear regression	0.64002	0.610000
Decision tree	0.70785	0.698334
Random forest ensemble	0.79312	0.801799
Gradient boosting ensemble	0.74109	0.718364

Table 11: Model evaluation for deriving identity affirmation index

5.3 PERFORMANCE METRICS

Performance statistics related data collected during the design and build of Inquest system components are presented.

5.3.1 ANALYTICS METHODS

This section outlines the execution time in seconds for the Sci-kit learn routines on the simulated data set. A 2009 model MacBook with 2.26GHz Intel Core 2 Duo P7550 processor and 2GB memory is used for performing the simulation. The specifics on the training and test dataset is presented in chapter 3. X is marked to represent not applicable combination sets.

Evaluation method	Location confidence	Usage confidence	Word confidence	Identity affirmation
Linear regression	0.093	0.181	X	0.071
Decision tree	0.129	0.186	X	0.089
Random forest ensemble (default estimators)	0.835	0.932	X	0.746
Gradient boosting ensemble (default estimators)	1.124	1.325	X	0.957
Support vector regression	X	X	1.834	X
Support vector machine with RBF kernel	X	X	2.045	X
Naïve Bayes	X	X	1.165	X

Table 12: Model performance on the simulated dataset.

5.4 SOFTWARE ENGINEERING METRICS

Inquest client service component is written in java, while server components are written in python.

5.4.1 VERSION CONTROL

The git revision control system [29] is used to manage the changes throughout the development cycle. The statistics given below were gathered at the time of completing this report from the repository holding all components of Inquest, scripts from design preparatory work, and this report.

Commits – 87, Insertions – 21,428 lines, Deletions – 9,192 lines

Chapter 6: Conclusion

This report looks into aspects of securing user identity on mobile devices making use of perceptible information readily available for that device. It proposes an instantiation of the envisioned system and outlines the design for the system components.

6.1 Lessons Learnt

When designing and developing rigorous data consuming applications with client server architecture, designer shall consider

- Abstracting the solution as a service which the client user can subscribe to.
- Using REST style API architecture helps scalable component interactions and generality of interfaces.
- Truth ness of the data set and its availability plays an important role in data critical predictive analytics applications.
- Giving enough importance for performance consequences in choosing web container technologies and associated tools and packages.

6.1.1 What went well

Simulation of training and test dataset from social media worked well to evaluate the models to be chosen for integration. It is expected that this information will closely match the kind of characteristics of information gathered in real time Inquest environment and shall be used as the training dataset until a much more reliable training dataset can be generated from a production like environment.

Choosing several relevant predictive models and evaluating them for their effectiveness making use of simulated training and test dataset helped to generate a view of model effectiveness and selection of a model for integration in server data processing environment.

Development of the Inquest client and server components utilizes several design patterns, which contributes in realizing a scalable and working system. Façade pattern is leveraged to define a cleaner interface between connected modules or classes. The façade interface exposes a controlled or simplified subset of functionality where the interface implementation makes use of calls into richer and complex subsystems. Database access is enforced through singleton implementation of the class to enforce serialization. The Django rest framework makes use of model view controller pattern separating out the inputs to application from the business processing logic and output format logic. Publisher subscriber and observer patterns are used to notify of state changes like data availability for further processing. The subscriber class contains a method that can be used by publisher class to notify the registered objects of any changes.

At the client service the proxy pattern shall be leveraged to provide a simplified interface. Complexity of the server data model and operations are not exposed. The definition of REST style API enabled to have an interface definition which is agnostic of tools and techniques used in underlying implementation. Façade pattern is leverage is have clear boundary for the separation of responsibilities across the software components.

The concept employed to average out the identity affirmation indices for the past N hours, N being an attribute of the quality of service registration parameter settings, shall help in normalizing the reported identity affirmation value. The client device is also programmed for a frequent information capture and identity affirmation checks on a negative tending identity affirmation value.

6.1.2 WHAT DID NOT GO WELL

Outlier detection techniques were not included in the initial thought process for the system design which if considered shall help to clean training and test dataset. Outlier

detection when combined with ability to obsolete old not any more relevant information from the dataset shall contribute to a cleaner dataset which can provide better prediction accuracy.

6.2 EXISTING TECHNOLOGIES IN THIS AREA

Existing technologies in the area of personal device theft detection and prevention are the applications which are used to track the personal device locations. Android device manager is such a kind of application which finds the approximate device location on a map and information on when it was last used. Such applications are mostly beneficial when a user tries to locate the device if the device are not turned off and still connected on network.

6.3 FUTURE WORK PLANS

The immediate priority shall be the completion of the client server prototype implementation to be ready for evaluation in a multi user multi device environment. It is expected that the effectiveness of the model shall improve as we gather much more wider and relevant real life data set. Methods to simulate and test scalability of the system in a much larger user environment shall be investigated.

Outlier detection techniques shall be employed on the dataset. Also methods shall be identified which can discard obsolete dataset from the model training set. Outlier detection combined with ability to obsolete old not any more relevant information from the dataset shall contribute to a cleaner dataset which can provide better prediction accuracy.

This report is not attempting to address the security considerations associated with transmitting the information fingerprints in a client server environment. There are number of artifacts on using private and public key technologies to enable a secure channel of

communication. HTTPS is popular technology which the solution shall be migrated over to enable a wider user adoption.

Ability to have a view of affirmation status summary through a web based console access shall be useful in the event of an alert and if user wants to track the device through an internet browser application.

References

[1] Mobile personal device proliferation statistics from Gartner - http://www.gartner.com/newsroom/id/2408515

[2] GPS - https://en.wikipedia.org/wiki/Geographic_coordinate_system

[3] REST based client server interface model and REST constraints: Fielding - http://www.ics.uci.edu/~fielding/pubs/dissertation/software_arch.htm

[4] REST introduction - http://www.restapitutorial.com/lessons/whatisrest.html#

[5] HTTP codes - http://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html

[6] Django Python Framework - https://www.djangoproject.com/

[7] Amazon web server Elastic Bean Stalk environment - http://docs.aws.amazon.com/elasticbeanstalk/latest/dg/Welcome.html

[8] Pandas Python data analysis library - http://pandas.pydata.org/

[9] Pandas tutorial - http://pandas.pydata.org/pandas-docs/stable/pandas.pdf

[10] Python Scikit learn library- http://scikit-learn.org/

[11] Stop words in English language - http://www.ranks.nl/stopwords

[12] Android developer reference documentation - https://developer.android.com/reference/android

[13] Amazon AWS Elastic Beanstalk concepts and architecture - http://docs.aws.amazon.com/elasticbeanstalk/latest/dg/concepts.concepts.architecture.html

[14] Android services - http://developer.android.com/guide/components/services.html

[15] Amazon S3 - https://aws.amazon.com/s3/

[16] Django rest framework - http://www.django-rest-framework.org/

[17] Graph Databases – New opportunities for connected data; Ian Robinson, Jim Webber and Emil Eifrem - http://info.neo4j.com/rs/neotechnology/images/Graph_Databases_2e_Neo4j.pdf

[18] Mongodb manual - https://docs.mongodb.org/manual/

[19] Amazon DynamoDB - https://aws.amazon.com/dynamodb/

[20] Redis data types - http://redis.io/topics/data-types-intro

[21] Redis Sentinel - http://redis.io/topics/sentinel

[22] Statistics and data analysis: Dell statsoft text book - http://www.statsoft.com/Textbook/

[23] Fundamentals of predictive text mining- (Sholom M. Weiss, Nitin Indurkhya, Tong Zhang)

[24] Decision Trees Scikit learn API reference - http://scikit-learn.org/stable/modules/tree.html

[25] Random Forests: Breiman - http://oz.berkeley.edu/~breiman/randomforest2001.pdf

[26] Gradient Boosting Models: Scikit learn API reference- http://scikitlearn.org/stable/modules/generated/sklearn.ensemble.GradientBoostingClassifier.html

[27] Support Vector Machines and Support Vector Regressions: Scikit learn documentation and API reference - http://scikit-learn.org/stable/modules/svm.html

[28] Naïve Bayes: Scikit learn documentation and API reference - http://scikit-learn.org/stable/modules/naive_bayes.html

[29] Git version control system documentation - https://git-scm.com/book/en/v2/Getting-Started-About-Version-Control